(12) UK Patent Application (19) GB (11) 2 043 311 A

(21) Application No **7907406**
(22) Date of filing **2 Mar 1979**
(43) Application published
1 Oct 1980
(51) **INT CL³**
G06F 7/00
(52) Domestic classification
G4H 13D 1A TA
(56) Documents cited
None
(58) Field of search
G4A
G4H
H4T
(71) Applicants
International Business Machines Corporation, Armonk, New York 10504, United States of America.
(72) Inventors
Michael Fredric Cowlishaw
(74) Agents
John E. Appleton

(54) **Text processing**

(57) A text processing system in which text is stored in a hierarchical tree structure form includes means to implement the forming of new blocks of text at any level in the tree structure and to move the display window up and down the tree structure in response to commands issued by a user at an interactive display terminal connected to a pr cessing unit.

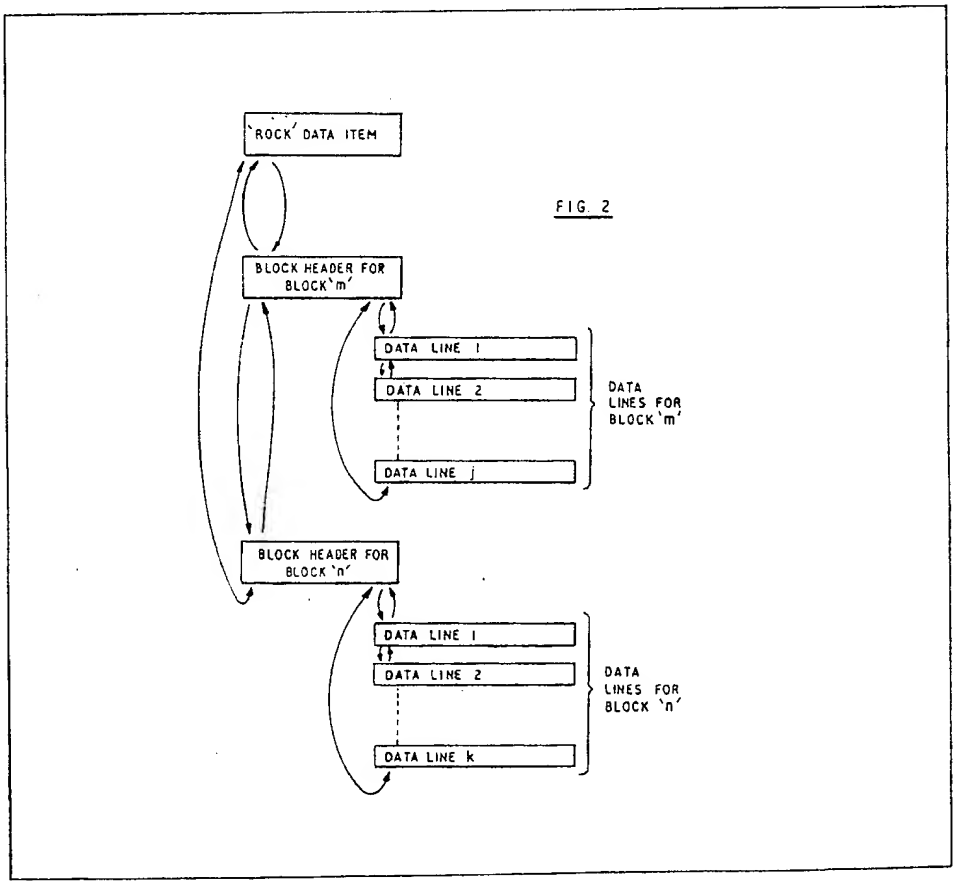

FIG 2

GB 2 043 311 A

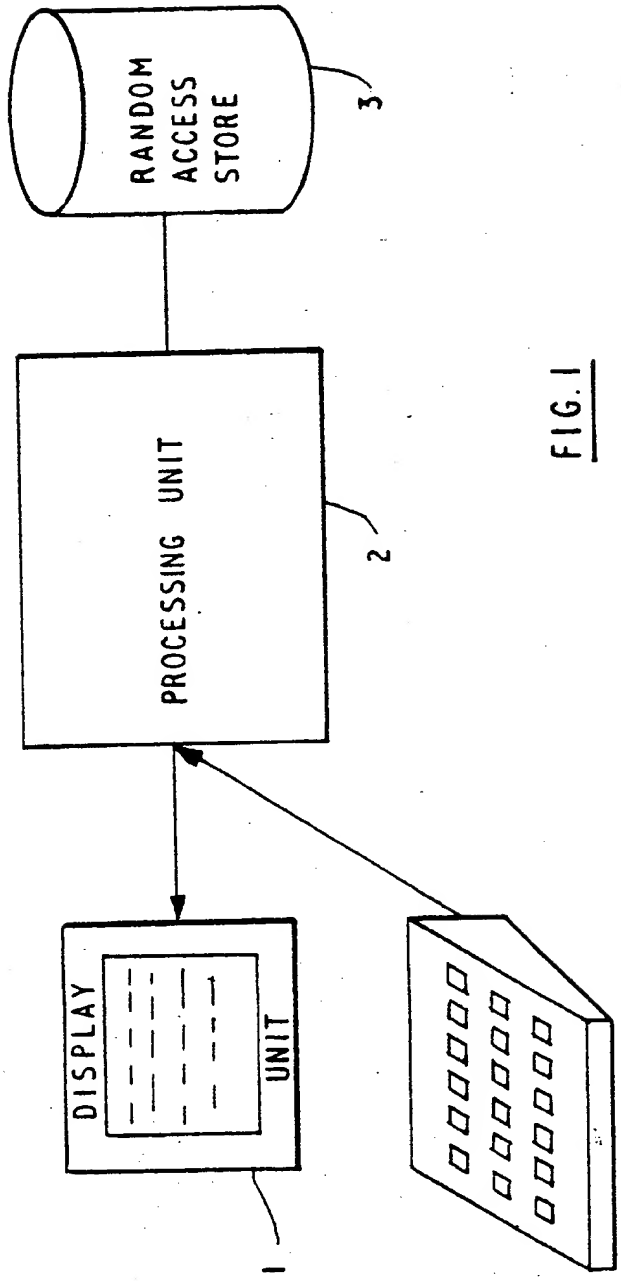

FIG.1

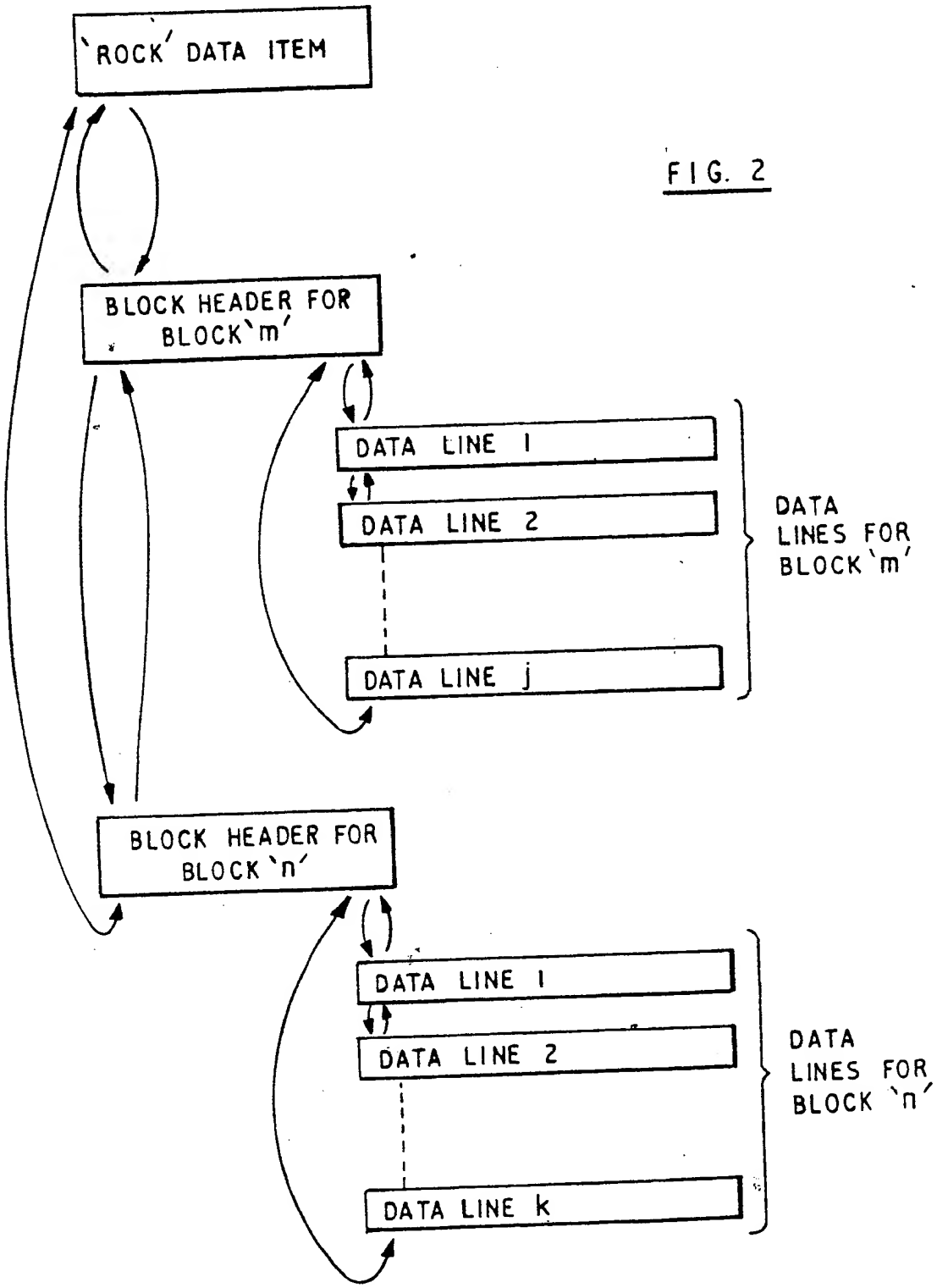
'ROCK' DATA ITEM
BLOCK HEADER FOR BLOCK 'm'
DATA LINE 1
DATA LINE 2
DATA LINE j
DATA LINES FOR BLOCK 'm'
BLOCK HEADER FOR BLOCK 'n'
DATA LINE 1
DATA LINE 2
DATA LINE k
DATA LINES FOR BLOCK 'n'

FIG. 2

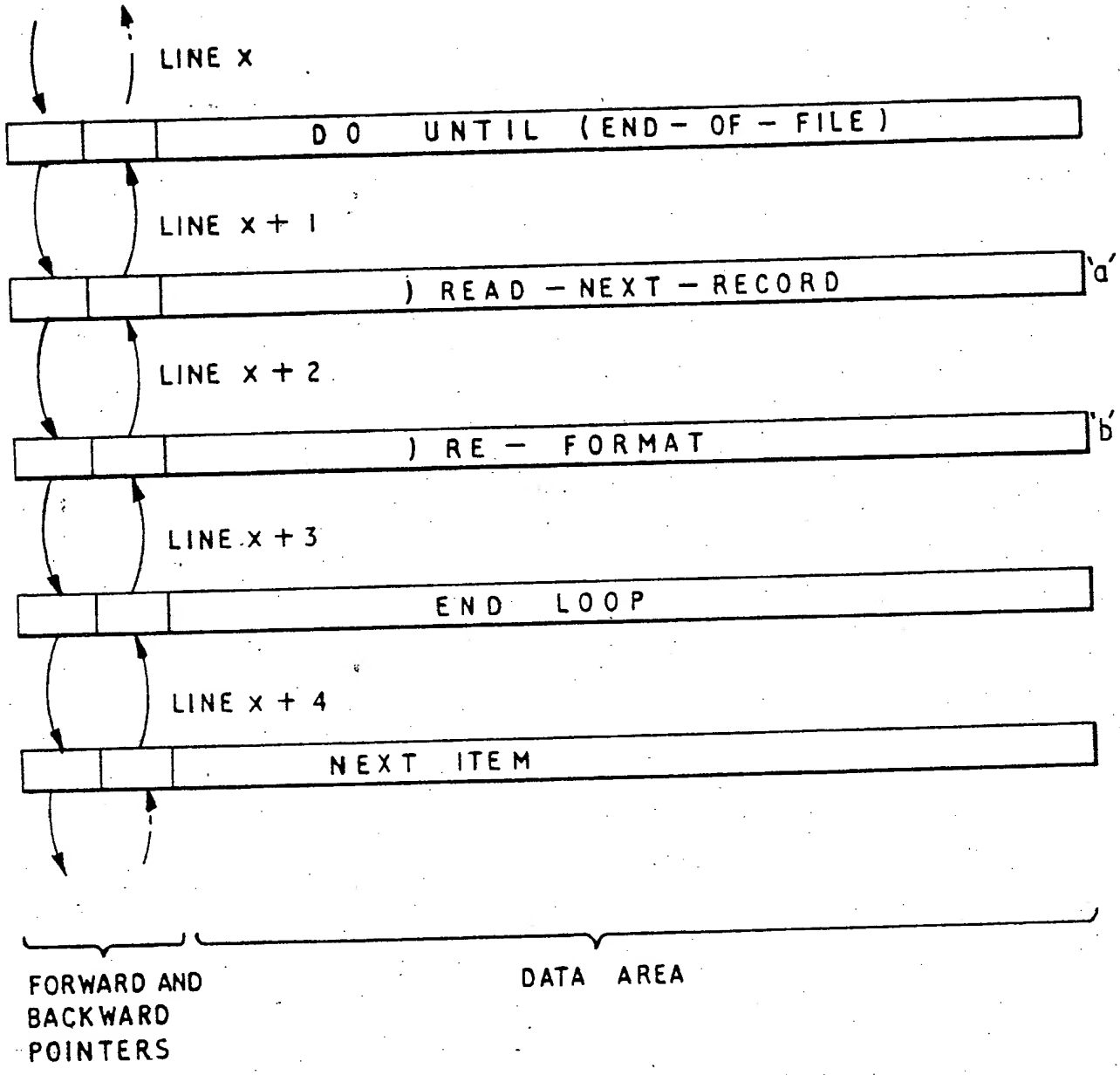
LINE X
DO UNTIL (END-OF-FILE)
LINE X + 1
) READ-NEXT-RECORD
'a'
LINE X + 2
) RE-FORMAT
'b'
LINE X + 3
END LOOP
LINE X + 4
NEXT ITEM
FORWARD AND BACKWARD POINTERS
DATA AREA

FIG. 3

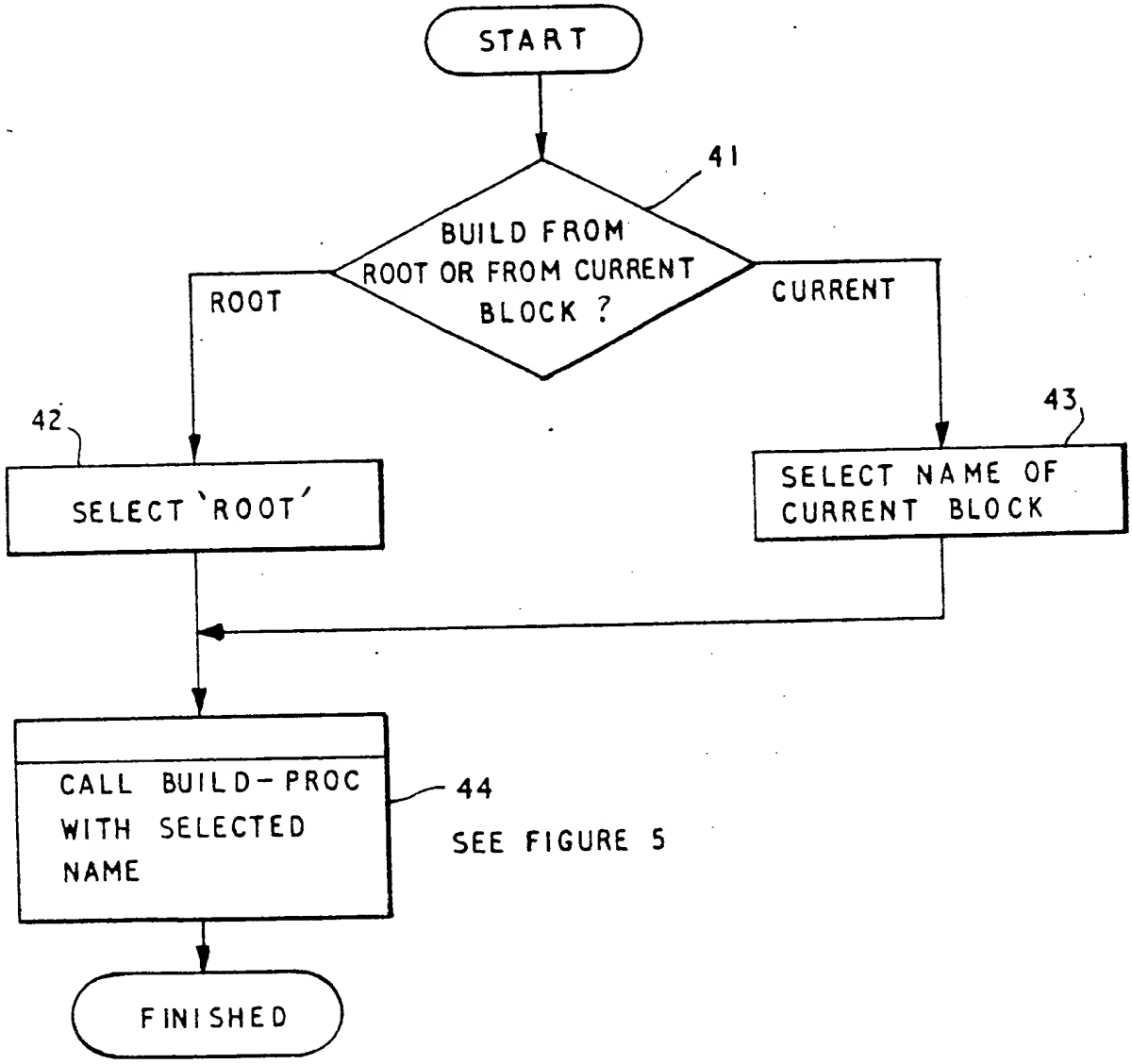
START
41
BUILD FROM ROOT OR FROM CURRENT BLOCK ?
ROOT
CURRENT
42
SELECT 'ROOT'
43
SELECT NAME OF CURRENT BLOCK
CALL BUILD-PROC WITH SELECTED NAME
44
SEE FIGURE 5
FINISHED

FIG. 4

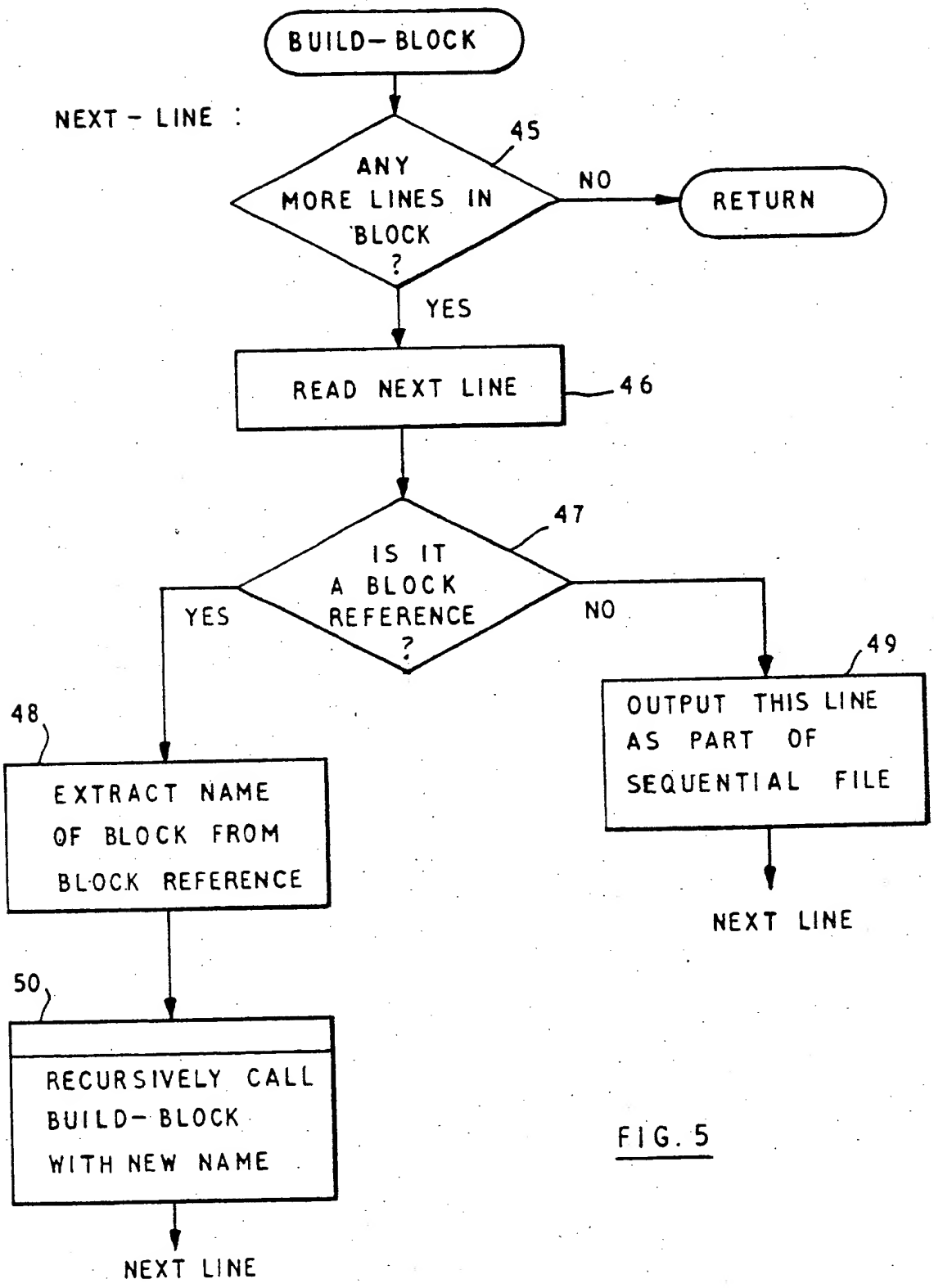
BUILD-BLOCK
NEXT-LINE :
45
ANY MORE LINES IN BLOCK ?
NO
RETURN
YES
READ NEXT LINE
46
47
IS IT A BLOCK REFERENCE ?
YES
NO
48
EXTRACT NAME OF BLOCK FROM BLOCK REFERENCE
49
OUTPUT THIS LINE AS PART OF SEQUENTIAL FILE
NEXT LINE
50
RECURSIVELY CALL BUILD-BLOCK WITH NEW NAME
NEXT LINE

FIG. 5

ROOT
- CHAPTER-1
  - CH.1, SECTION-A
  - CH.1, SECTION-B
  - CH.1, SECTION-C
    - FIGURE-1
    - FIGURE-2
- CHAPTER-2
  - CH.2, SECTION-A
    - Table-1
    - FIGURE-3
  - CH.2, SECTION-B
    - FIGURE-3
    - Table-2
- SUMMARY
- APPENDIX-A
- APPENDIX-B

FIG 6

SPECIFICATION

Improvements in text processing systems

This invention relates to text processing systems which include a central processing system and at least one interactive display terminal and particularly to such systems adapted to perform editing functions.

The ability of data processing system to store and manipulate data has already been exploited in the field of text processing. An author may use an interactive display terminal to enter text into a data processing system which displays the text on a display device such as a cathode ray tube and when the author is satisfied, the text is stored in a memory device which may, for example, be a magnetic disc store. In order to manipulate the data, the system must be adapted to perform editing functions. Editor programs are now well-known in the field of data processing and a comprehensive survey of editors is found in the paper On-line Text Editing: A Survey by Andries van Dam and David E. Rice in Computing Surveys, Vol. 3, No. 3, Sept. 1971, pp 93-114. The paper describes several different approaches to editor programs, the most relevant to the present invention being an editor called NLS developed at Stanford University. It is pointed out in the paper that "Of particular significance is the ability of NLS to display a file from many different points of view. For example, the hierarchical outline structure of a text - the various headings - may be stored as part of the data structure, and one may ask to see, for example, only section 3.1.4, or all sections down to two levels of subsections, or the first line in each of the subsections on the fourth level. The text is thus viewed as a collection of sections (called "statements") with a tree structure superimposed on this basic data structure. Each statement (less than 3000 characters) is meant to contain a single complete thought or idea, but may have substatements down to an arbitrary number of levels. Most standard tree manipulations are allowed at a given level in the tree; e.g., locating or deleting the next node or the previous one, locating the first subnode, rearranging neighbouring nodes, etc." This hierarchical approach to files is useful for documents and for editing computer programs.

It should be noted that in the NLS approach to editors the text has a structure imposed upon it by the program. It is an object of the present invention to provide an editor in which the data is self-structuring and in which the structure can be changed by changing the data.

UK Patent Specification 1,363,910 describes a text manipulating processing system in which a text may be edited by using a series of symbolic variables and in which the lines of data or text are presented to the user as a continuous scroll. It is a further object of the present invention to provide a structured editor program that permits a file of data to be broken down into a series of blocks of lines which are essentially self indexing and self documenting and which form a tree structure which the user may traverse at will.

A common concept in the field of data processing is that of a 'file' of data which may hold numerical data, computer programs, documentation (text), as well as other information. If it is required to make a change to a file of data it is necessary to use a program performed by the computer to access the file and allow the user to effect the changes, such as alternating lines, copying groups of lines, moving data around the file etc. Usually the file is presented to the user as though it is a long scroll or block, which may be many thousands of lines in length (see UK Patent 1,363,910).

A major limitation of the productivity of the user editing the file is the time taken to locate a position in a file, since in a large file (more than a few hundred lines) it is difficult to maintain an awareness of the current position in the file and to keep a sense of context. For example, in editing a document which consists of a number of chapters, the chapters are normally presented one after the other and the user has no easy way of telling exactly where he is editing if it is not obvious from the few lines presented on the display device.

Similarly a computer program has a logical structure which is known by the author of the program: however this structure does not normally physically exist in the file due to the sequential nature of the file itself. If the logical structure was mirrored by a physical structure in the file, the structure would be immediately apparent to any person who accesses the file (not only the author), and lower levels of programming could be by-passed when scanning the file in order to edit it.

It is therefore a further object of the invention to provide a text processing system in which a file of data can be broken down into logical blocks which can then be independently accessed by a user, while maintaining the interdependencies of the blocks and the physical integrity of the file.

According to the invention, there is provided, a text processing system including a processing unit and at least one interactive display terminal, the processing unit including a main store and processing means adapted to perform a text editing process under control of commands received from the interactive display terminal, the text being stored in the form of blocks of text lines, each block being identified by a header record in a list of header records and associated with other blocks in a hierarchical tree structure so that at least one record in a high level block includes an identification of a next lower level block and including means to form a new block from text data entered at the interactive display terminal by adding a new block header record to the list of header records and inserting in the new block header record a pointer to the storage position of the first and the last line of the associated text data.

In order that the invention may be fully understood a preferred embodiment thereof will now be described with reference to the accompanying drawings, in which:

Figure 1 is a schematic diagram of a text processing system,

Figure 2 shows the data structure within the text

processing system of the present invention,

Figure 3 shows part of a sample list of basic data,

Figures 4A and *B* illustrate a method used in the text processing system, and

Figure 5 shows a display of file structure.

The structural editing data processing system allows the user to manipulate groups of lines (called 'Blocks') as easily as the other editors permit the manipulation of single lines. A multiple level tree structure of blocks may be naturally and easily created, either from the time of file creation or, if necessary, by structuring an existing sequential file. When required, a conventional file may be 'built' from the structured file in order to be processed by other routines in the system (such as compilers, text formatters, assemblers) which cannot directly handle structured files.

In the preferred embodiment of the invention the text processing system allows the user access to a structured file (in addition to conventional sequential files) and the permits the user to edit the various component blocks of the file using a conventional display screen together with a keyboard, such as the IBM 3277 (IBM is a Registered Trade Mark). The display screen acts as a 'window' into a small portion of the file. The keyboard is used for entering data and also various commands which manipulate the data or blocks in the file and adjust the window into the file as seen by the user on the display screen. Most of the normal editing commands such as those for writing the file on to a magnetic disk, locating text strings, etc., are implemented. However the present invention is only concerned with those that deal with the structure of the file and these will be described below.

Referring now more particularly to the drawings, Figure 1 shows in schematic form the hardware required for a basic text processing system. An interactive terminal 1, comprising a display screen and a keyboard is connected to a processing unit 2. The processing unit, which may be a machine such as an IBM 370/168 processor, includes an operating system and system control programs which enable a user of the terminal 1 to have interactive communication with an application program being run in the processor. Application programs and data files are stored in an auxilliary memory 3 which is connected to the processor 2. The auxilliary memory may be a magnetic disc store such as is commonly used with data processing systems. Of course a processor such as an IBM 370/168 will support many interactive terminals simultaneously and the connection may be over many kinds of communication link. There will also be several types of auxilliary store connected to the main processing unit. However the processing ability of the processing unit is not directly concerned with the present invention and it is possible, with the advances of today's technology, to envisage the invention being embodied in a system contained in a single unit incorporating all three components of Figure 1.

The preferred embodiment of the text processing system of the present invention will now be described with reference to Figures 2 - 5.

A structured file is held in the auxilliary store 3 (Figure 1) e.g., a magnetic disk, in a special format which is then read into the main store of the processing unit 2 when a user at terminal 1 desires to edit this file.

As shown in Figure 2 the basic data forming the file is then held as a series of lists (1 - j and 1 - k; one list for each block of lines) which in turn are descendants of a single linked list of data items, each of which comprises a 'header record' for one block (M, N) of lines. The list of header records is itself dependant on a fixed item of data termed the 'Rock' which also contains basic information about the file (such as the format of the file, the current tabulation settings, etc.).

For efficiency, an alphabetically sorted index to the block is also constructed by the structured editing program: this allows a rapid binary search for blocks to be made when required.

Figure 3 shows several consecutive lines of data in this particular example part of the text of a computer program. In it most of the lines (x through x + 4) are 'true data' i.e., they are lines of the program. Lines 'a' and 'b' however are 'block references': these are distinguished by a special character selected by the user (in this case ")") being the first non-blank character of the line.

The following non-blank characters (up to 16 in the preferred embodiment) form the name of the block that is being referenced. Each block reference refers to another block in the file which therefore appears to be at a lower level from the user's point of view. The block references therefore both define the structure of the file and act as an automatic index to the tree structure. Since the block references are held in the same way as data lines they can be altered as easily as the data itself. Any block may contain any number of block references, or may even consist solely of block references. A powerful and very flexible multi-level tree structure can be set up and manipulated easily by the user.

An important facility is the ability to 'Form' a block from a previously selected group of lines. First a new block header is added to the list of headers, then the selected group of lines is moved to form the new block, and finally a block reference to the new block is inserted into the original block at the point where the group of lines used to be. This facility makes it possible to quickly structure an existing sequential file or reduce the size of blocks that have grown too large.

Three commands, usually assigned a special function key on the keyboard, allow the user to traverse the tree structure that has been built with the minimum of effort. The first of these commands is 'Down' which carries out the following operation: The current block is scanned, from the top of the current window, for any block references. If one is found (nothing happens otherwise), the block named in the block reference becomes the current block, i.e., the top of the new block becomes the window into the file. The 'Down' command therefore has the effect of moving down a level in the file. Each time a 'downwards' movement is made in the file the movement is recorded by the text processing system, so building up a history of the path the user

used to get to the current block. This path can then be retraced by issuing a second traversing command 'Back', which therefore is only valid following a 'Down' command. The 'Back' command has the effect of moving back up a level, in that it takes the user from the current block up to the block above, e.g., the block from which the current block was entered. The current window is set to the line below the block reference that was used when the last 'Down' command was issued. The third traversing command which is especially useful is 'Next'. This has the same effect as a Back command followed by a Down command and takes the user to the next block at the same level as the current block.

Because most processing systems do not, as yet, accept structured data files directly, it is necessary to be able to produce a sequential file from the structured data. A 'build' command makes the text processing system scan the file (either from the top of the tree structure, the root, or from the current block) and replace all block references at all levels with the data lines from the referenced blocks. This generates a conventional sequential file without any block references. The method used to accomplish this build procedure is shown in Figures 4A and 4B. Figure 4A shows a flow chart of the build method in general form. Initially a decision is taken at Step 41 whether to build from the 'ROOT' or from the current block. If from the ROOT the root is selected at Step 42, if from the current block, the name of the current block is selected at Step 43. This is then followed by the Step 44 which is illustrated in more detail in Figure 4B.

Step 44 'Call Build Process with Selected Name' comprises the steps shown in Figure 4B starting at Step 45 in which a decision is taken as to whether there are any more lines in the block. If not, the routine returns to Step 44 and Finishes. If there are, then at Step 46 the next line is read. Step 47 is then a decision step as to whether what has been read is a block reference or not? If not the line is output at Step 48 as part of the sequential file and the next line is read. If it is a block reference then at Step 48 the name is extracted from the block reference and the new build block is recursively called at Step 50 to re-enter the routine at Step 45.

Since the text editing processing system contains information about the structure of the data file because the structure is part of the data, the structure may be shown to the user on demand. A 'List' command causes a listing of the blocks to be displayed indenting according to their level in the file. Figure 5 shows an example of such a display. The listing can be of all the file, part of the file, or just a certian number of levels in the file. The process used to generate the structure display is identical to that used for Build (Figures 4A and B), except that instead of outputting all data lines encountered during the search, the process just displays the name of each block as it is entered during the search.

A text editing processing system has been described above embodying the present invention, it is apparent that there may be other features used in an editing system which have not been described, but are not required for an understanding of the invention.

CLAIMS

1. A text processing system including a processing unit and at least one interactive display terminal, the processing unit including a main store and processing means adapted to perform a text editing process under control of commands received from the interactive display terminal, the text being stored in the form of blocks of text lines, each block being identified by a header record in a list of header records and associated with other blocks in a hierarchical tree structure so that at least one record in a high level block includes an identification of a next lower level block and including means to form a new block from text data entered at the interactive display terminal by adding a new block header record to the list of header records and inserting in the new block header record a pointer to the storage position of the first and the last line of the associated text data.

2. A text processing system as claimed in claim 1 including means to display a block of data text at the interactive display terminal and means to scan the current block for a reference to a lower level block header reference and, when one is found, to replace the block currently being displayed with the block currently being displayed with the block of text associated with the lower level block header.

3. A text processing system as claimed in claim 2 including means to store the route of associated blocks through which a user reaches a currently displayed block and means to retrace the route so that a currently displayed block is replaced by the previously displayed block of text data.

4. A text processing system as claimed in claim 3 in which the means to scan the current block for a reference to a lower level block stores the first occurrence of a lower level block header reference and in response to a command issued at the interactive display terminal scans the current block for further lower level block header references and replaces the current block with the block associated with the next block header reference found.

5. A text processing system as claimed in any one of claims 1 to 4 including means to construct a sequential file from a set of text data stored in a hierarchical tree structure form.

6. A text processing system substantially as hereinbefore described with reference to, and as illustrated in, Figures 2 to 5 of the accompanying drawings.

Printed for Her Majesty's Stationery Office by Croydon Printing Company Limited, Croydon Surrey, 1980.
Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.